The Handbook of Brain Theory and Neural Networks

EDITED BY
Michael A. Arbib

A Bradford Book
THE MIT PRESS
Cambridge, Massachusetts
London, England

This book was set in Times Roman by Asco Trade Typesetting Ltd., Hong Kong, and was printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

The handbook of brain theory and neural networks / Michael A. Arbib, editor.
p. cm.
"A Bradford book."
Includes bibliographical references and index.
ISBN 0-262-01148-4
1. Neural networks (Neurobiology)—Handbooks, manuals, etc.
2. Neural networks (Computer science)—Handbooks, manuals, etc.
I. Arbib, Michael A.
QP363.3.H36 1995
612.8'2—dc20 94-44408
CIP

poor scaling behavior somewhat dampened this enthusiasm, and this behavior was traced to the use of soft constraints. The next generation of networks, using hard constraints, seemed to scale well (Peterson, 1990). Moreover, it also seems possible for algorithms with soft constraints to scale well, provided attention is paid to the values of the coefficients in the bias terms (Aiyer et al., 1990).

Neural network optimization has developed into a good general-purpose heuristic algorithm that is competitive with other optimization procedures. On well-studied problems like the TSP, however, it still remains inferior to the best computer science algorithms.

There is some difficulty, however, in comparing neural network algorithms fairly to existing computer algorithms. The neural network algorithms are designed to work on special-purpose analog hardware, and they suffer when these equations are discretized for serial computation. A new generation of neural network algorithms—discrete in time, though parallel and using analog variables (Waugh and Westervelt, 1993)—may be more directly comparable.

Finally, recent theoretical analysis (Yuille and Kosowsky, 1994) shows some precise relations between neural network optimization and more traditional optimization methods using barrier functions and interior point methods (Gill et al., 1986).

Appendix: Desirable Background Knowledge

The mean-field equations can be obtained in at least three different ways (see Reif, 1982, and Hertz et al., 1991):

1. *The saddle-point approximation.* This is a standard technique in statistical mechanics. The partition function $Z = \Sigma_V e^{-\beta E[V]}$ is transformed into an integral form $Z = \int [dS][dU] e^{-\beta E_{\text{eff}}[S,U;\beta]}$, where the S variables correspond to the mean fields of the V and the U are auxiliary variables which impose constraints on the S. $E_{\text{eff}}[S, U : \beta]$ is extremized with respect to S and U; then the mean fields are approximated by the resulting values S^*.
2. *Approximating the free energy.* The free energy $F = -T \log Z$. From thermodynamic relations, it can be expressed as $U - TS$, where U is the energy and S is the entropy. The expectation of the free energy, with respect to the Gibbs distribution, is approximated to give the effective energy.
3. *Mean-field techniques.* When evaluating the partition function, one evaluates the contribution from each variable by fixing the remaining variables at their mean values. This leads to consistency conditions for the mean fields, the mean-field equations.

Note that physicists have established conditions under which these approximations are valid. Unfortunately, these situations typically involve making the system infinitely large and imposing undesirable conditions on the form of the energy function. For our purposes, however, we are interested only in the weaker result that these approximations are valid at zero temperature.

Acknowledgments. I would like to thank DARPA and the Air Force for support under contract F49620–92–J–0466.

Road Map: Dynamic Systems and Optimization
Background: I.3. Dynamics and Adaptation in Neural Networks
Related Reading: Energy Functions for Neural Networks; Statistical Mechanics of Neural Networks

References

Aiyer, S. V. B., Niranjan, M., and Fallside, F., 1990, A theoretical investigation into the performance of the hopfield model, *IEEE Trans. Neural Netw.*, 1:204–215.

Durbin, R., and Willshaw, D., 1987, An analog approach to the travelling salesman problem using an elastic net method, *Nature*, 326:689–691.

Durbin, R., Szeliski, R., and Yuille, A. L., 1989, An analysis of the elastic net approach to the travelling salesman problem, *Neural Computat.*, 1:348–358.

Gill, P., Murray, W., Saunders, M., Tomlin, J., and Wright, M., 1986, On projective newton barrier methods for linear programming and an equivalence to Karmarkar's projective method, *Math. Programming*, 36:183–209.

Hertz, J., Krogh, A., and Palmer, R. G., 1991, *Introduction to the Theory of Neural Computation*, Redwood City, CA: Addison-Wesley.

Hopfield, J. J., and Tank, D. W., 1985, Neural computation of decisions in optimization problems, *Biol. Cybern.*, 52:141–152.

Kirkpatrick, S., Gelatt, C., Jr., and Vecchi, M., 1983, Optimization by simulated annealing, *Science*, 220:671–680.

Peterson, C., 1990, Parallel distributed approaches to combinatorial optimization problems—benchmark studies on T.S.P., *Neural Computat.*, 2:261–270.

Reif, F., 1982, *Fundamentals of Statistical and Thermal Physics*, New York: McGraw-Hill.

Simic, P., 1990, Statistical mechanics as the underlying theory of "elastic" and "neural" optimization, *Network: Computat. Neural Syst.*, 1:1–15.

Waugh, F., and Westervelt, R., 1993, Stability of analog networks with local competition, *Phys. Rev. E* (in press).

Yuille, A. L., 1990, Generalized deformable models, statistical physics and matching problems, *Neural Computat.*, 2:1–24.

Yuille, A. L., and Kosowsky, J. J., 1994, A mathematical analysis of deterministic annealing, in *Artificial Neural Networks with Applications in Speech and Vision*, New York: Chapman and Hall.

Yuille, A. L., Stolorz, P., and Utans, J., 1994, Statistical physics, mixtures of distributions, and the EM algorithm, *Neural Computat.*, 6:334–340.

Convolutional Networks for Images, Speech, and Time Series

Yann LeCun and Yoshua Bengio

Introduction

The ability of multilayer backpropagation networks to learn complex, high-dimensional, nonlinear mappings from large collections of examples makes them obvious candidates for image recognition or speech recognition tasks (see PATTERN RECOGNITION). A typical approach is to feed the network with "raw" inputs (e.g., normalized images) and to rely on backpropagation to turn the first few layers into an appropriate feature extractor. While this can be done with an ordinary fully connected feedforward network with some success for tasks such as character recognition, there are problems.

First, typical images, or spectral representations of spoken words, are large, often with several hundred variables. A fully

connected first layer with, say, a few hundred hidden units, would already contain tens of thousands of weights. Overfitting problems may occur if training data are scarce. In addition, the memory requirement for that many weights may rule out certain hardware implementations. But the main deficiency of unstructured nets for image or speech aplications is that they have no built-in invariance with respect to translations, or local distortions of the inputs. Before being sent to the fixed-size input layer of a neural net, character images, spoken word spectra, or other 2D or 1D signals must be approximately size normalized and centered in the input field. Unfortunately, no such preprocessing can be perfect: handwriting is often normalized at the word level, which can cause size, slant, and position variations for individual characters; words can be spoken at varying speed, pitch, and intonation. This causes variations in the position of distinctive features in input objects. In principle, a fully connected network of sufficient size could learn to produce outputs that are invariant with respect to such variations. However, learning such a task would probably result in multiple units with identical weight patterns positioned at various locations in the input. Learning these weight configurations requires a very large number of training instances to cover the space of possible variations. Conversely, in convolutional networks (which will be defined in the next section), shift invariance is automatically obtained by forcing the replication of weight configurations across space.

Second, a deficiency of fully connected architectures is that the topology of the input is entirely ignored. The input variables can be presented in any (fixed) order without affecting the outcome of the training. On the contrary, images, or spectral representations of speech, have a strong 2D local structure, and time series have a strong 1D structure: variables (or pixels) that are spatially or temporally nearby are highly correlated. Local correlations are the reasons for the well-known advantages of extracting and combining *local* features before recognizing spatial or temporal objects. Convolutional networks force the extraction of local features by restricting the receptive fields of hidden units to be local.

Convolutional Networks

Convolutional networks combine three architectural ideas to ensure some degree of shift and distortion invariance: local receptive fields, shared weights (or weight replication), and, sometimes, spatial or temporal subsampling. A typical convolutional network for recognizing characters is shown in Figure 1 (from LeCun et al., 1990). The input plane receives images of characters that are approximately size normalized and centered. Each unit of a layer receives inputs from a set of units located in a small neighborhood in the previous layer. The idea of connecting units to local receptive fields on the input goes back to the perceptron in the early 1960s, and was almost simultaneous with Hubel and Wiesel's discovery of locally sensitive, orientation-selective neurons in the cat's visual system. Local connections have been reused many times in neural models of visual learning (see Mozer, 1991; LeCun, 1986; and NEOCOGNITRON). With local receptive fields, neurons can extract elementary visual features such as oriented edges, endpoints, or corners (or similar features in speech spectrograms). These features are then combined by the higher layers. As stated earlier, distortions or shifts of the input can cause the position of salient features to vary. In addition, elementary feature detectors that are useful on one part of the image are likely to be useful across the entire image. This knowledge can be applied by forcing a set of units, whose receptive fields are located at different places on the image, to have identical weight vectors (Rumelhart, Hinton, and Williams, 1986). The outputs of such a set of neurons constitute a *feature map*. At each position, different types of units in different feature maps compute different types of features. A sequential implementation of this, for each feature map, would be to scan the input image with a single neuron that has a local receptive field and to store the states of this neuron at corresponding locations in the feature map. This operation is equivalent to a convolution with a small-size kernel, followed by a squashing function. The process can be performed in parallel by implementing the feature map as a plane of neurons that *share* a single weight vector. Units in a feature map are constrained to perform the same operation on different parts of the image. A convolutional layer is usually composed of several feature maps (with different weight vectors), so that multiple features can be extracted at each location. The first hidden layer in Figure 1 has four feature maps with 5×5 receptive fields. Shifting the input of a convolutional layer will shift the output but will leave it unchanged otherwise. Once a feature has been detected, its exact location becomes less important, as long as its approximate position relative to other features is preserved. Therefore, each convolutional layer is followed by an additional layer which performs a local averaging and a subsampling, reducing the resolution of the feature map, and reducing the sensitivity of the output to shifts and distortions. The second hidden layer in Figure 1 performs 2×2 averaging and subsampling, followed by a trainable coefficient, a trainable bias, and a sigmoid. The trainable coefficient and bias control the effect of the squashing nonlinearity (for example, if the coefficient is small, then the neuron operates in a quasi-linear mode). Successive layers of convolutions and subsampling are typically alternated, resulting in a *bi-pyramid*: at each layer, the number of feature maps is increased as the spatial resolution is decreased. Each unit in the third hidden layer in Figure 1 may have input connections from several feature maps in the previous layer. The convolution/subsampling combination, inspired by Hubel and Wiesel's notions of "simple"

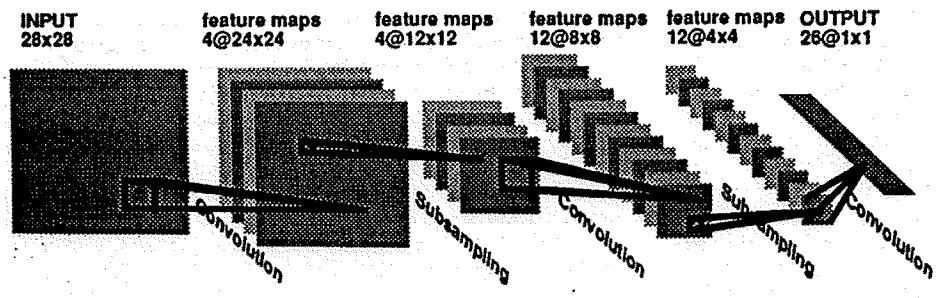

Figure 1. Convolutional neural network for image processing, e.g., handwriting recognition.

and "complex" cells, was implemented in the neocognitron model (see NEOCOGNITRON), although no globally supervised learning procedure such as backpropagation was available then.

Since all the weights are learned with backpropagation, convolutional networks can be seen as synthesizing their own feature extractor. The weight-sharing technique has the interesting side effect of reducing the number of free parameters, thereby reducing the "capacity" of the machine and improving its generalization ability (see LeCun, 1989, on weight sharing, and LEARNING AND GENERALIZATION: THEORETICAL BOUNDS for an explanation of generalization). The network in Figure 1 contains approximately 100,000 connections, but only approximately 2600 free parameters because of the weight sharing. Such networks compare favorably with other methods on handwritten character recognition tasks (Bottou et al., 1994; see also HANDWRITTEN DIGIT STRING RECOGNITION), and they have been deployed in commercial applications.

Fixed-size convolutional networks that share weights along a single temporal dimension are known as Time-Delay Neural Networks (TDNNs). TDNNs have been used in phoneme recognition (without subsampling) (Lang and Hinton, 1988; Waibel et al., 1989), spoken word recognition (with subsampling) (Bottou et al., 1990), and on-line handwriting recognition (Guyon et al., 1991).

Variable-Size Convolutional Networks: SDNNs

While characters or short spoken words can be size normalized and fed to a fixed-size network, more complex objects such as written or spoken words and sentences have inherently variable size. One way of handling such a composite object is to segment it heuristically into simpler objects that can be recognized individually (e.g., characters, phonemes). However, reliable segmentation heuristics do not exist for speech or cursive handwriting. A brute-force solution is to scan (or replicate) a recognizer at all possible locations across the input. While this can be prohibitively expensive in general, convolutional networks can be scanned or replicated very efficiently over large, variable-size input fields. Consider one instance of a convolutional net and its *alter ego* at a nearby location. Because of the convolutional nature of the networks, units in the two nets that look at identical locations on the input have identical outputs; therefore, their output does not need to be computed twice. In effect, replicating a convolutional network can be done simply by increasing the size of the field over which the convolutions are performed, and replicating the output layer, effectively making it a convolutional layer. An output whose receptive field is centered on an elementary object will produce the class of this object, while an in-between output may be empty or contain garbage. The outputs can be interpreted as evidence for the categories of object centered at different positions of the input field. A postprocessor therefore is required to pull out consistent interpretations of the output. Hidden Markov models (HMMs) or other graph-based methods are often used for that purpose (see SPEECH RECOGNITION: PATTERN MATCHING, and PATTERN RECOGNITION in this volume). The replicated network and the HMM can be trained simultaneously by backpropagating gradients through the HMM. Globally trained, variable-size TDNN/HMM hybrids have been used for speech recognition (see PATTERN RECOGNITION for a list of references) and on-line handwriting recognition (Schenkel et al., 1993). Two-dimensional replicated convolutional networks, called *Space Displacement Neural Networks* (SDNNs) (Figure 2), have been used in combination with HMMs or other elastic matching methods for handwritten word recognition (Keeler, Rumelhart, and Leow, 1991; Matan et al., 1992; Bengio, LeCun, and Henderson, 1994). Another interesting application of SDNNs is object spotting (Wolf and Platt, 1994).

An important advantage of convolutional neural networks is the ease with which they can be implemented in hardware. Specialized analog/digital chips have been designed and used in character recognition and in image-preprocessing applications (Boser et al., 1991). Speeds of more than 1000 characters per second were obtained with a network with approximately 100,000 connections.

The idea of subsampling can be turned around to construct networks that are similar to TDNNs but can generate sequences from labels. These networks are called reverse-TDNNs because they can be viewed as upside-down TDNNs: temporal resolution increases from the input to the output, through alternated oversampling and convolution layers (Simard and LeCun, 1992).

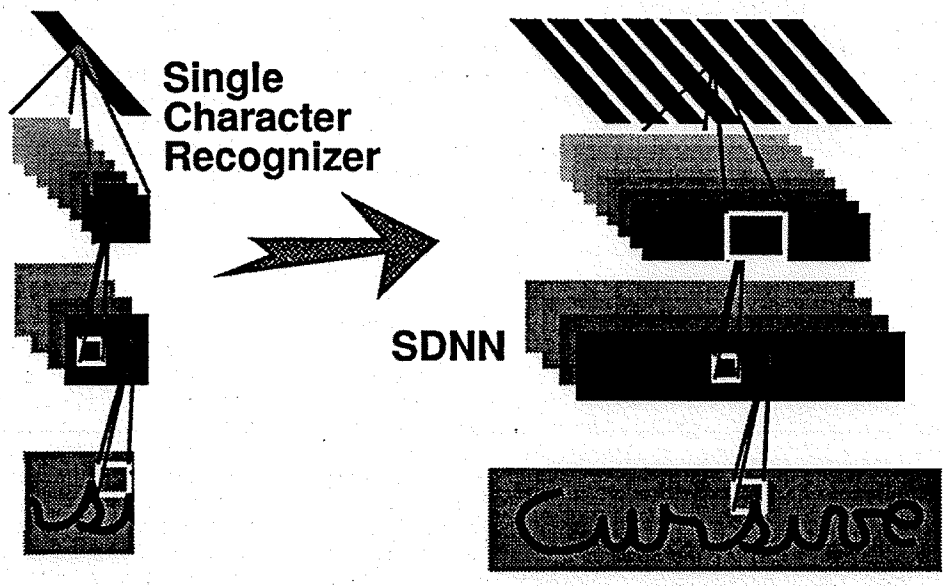

Figure 2. A variable-size replicated convolutional network, called a Space Displacement Neural Network (SDNN).

Discussion

Convolutional neural networks are a good example of an idea inspired by biology that resulted in competitive engineering solutions that compare favorably with other methods (Bottou et al., 1994). Although applying convolutional nets to image recognition removes the need for a separate hand-crafted feature extractor, normalizing the images for size and orientation (if only approximately) is still required. Shared weights and subsampling bring invariance with respect to small geometric transformations or distortions, but fully invariant recognition is still beyond reach. Radically new architectural ideas, possibly suggested by biology, will be required for a fully neural image or speech recognition system.

Road Map: Learning in Artificial Neural Networks, Deterministic
Background: I.3. Dynamics and Adaptation in Neural Networks

References

Bengio, Y., LeCun, Y., and Henderson, D., 1994, Globally trained handwritten word recognizer using spatial representation, space displacement neural networks and hidden Markov models, in *Advances in Neural Information Processing Systems 6* (J. Cowan, G. Tesauro, and J. Alspector, Eds.), San Mateo, CA: Morgan Kaufmann, pp. 937–944.

Boser, B., Sackinger, E., Bromley, J., LeCun, Y., and Jackel, L., 1991, An analog neural network processor with programmable topology, *IEEE J. Solid-State Circuits*, 26:2017–2025.

Bottou, L., Cortes, C., Denker, J., Drucker, H., Guyon, I., Jackel, L., LeCun, Y., Muller, U., Sackinger, E., Simard, P., and Vapnik, V., 1994, Comparison of classifier methods: A case study in handwritten digit recognition, in *Proceedings of the International Conference on Pattern Recognition*, Los Alamitos, CA: IEEE Computer Society Press.

Bottou, L., Fogelman-Soulié, F., Blanchet, P., and Lienard, J. S., 1990, Speaker independent isolated digit recognition: Multilayer perceptrons vs dynamic time warping, *Neural Netw.*, 3:453–465.

Guyon, I., Albrecht, P., LeCun, Y., Denker, J. S., and Hubbard, W., 1991, Design of a neural network character recognizer for a touch terminal, *Pattern Recognition*, 24(2):105–119.

Keeler, J., Rumelhart, D., and Leow, W., 1991, Integrated segmentation and recognition of hand-printed numerals, in *Advances in Neural Information Processing Systems 3* (R. P. Lippman, J. M. Moody, and D. S. Touretzky, Eds.), San Mateo, CA: Morgan Kaufmann, pp. 557–563.

Lang, K., and Hinton, G., 1988, *The Development of the Time-Delay Neural Network Architecture for Speech Recognition*, Technical Report CMU–CS–88–152, Carnegie-Mellon University, Pittsburgh.

LeCun, Y., 1986, Learning processes in an asymmetric threshold network, in *Disordered Systems and Biological Organization* (E. Bienenstock, F. Fogelman-Soulié, and G. Weisbuch, Eds.), Les Houches, France: Springer-Verlag, pp. 233–240.

LeCun, Y., 1989, *Generalization and Network Design Strategies*, Technical Report CRG–TR–89–4, Department of Computer Science, University of Toronto.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L., 1990, Handwritten digit recognition with a backpropagation network, in *Advances in Neural Information Processing Systems 2* (D. Touretzky, Ed.), San Mateo, CA: Morgan Kaufmann, pp. 396–404.

Matan, O., Burges, C., LeCun, Y., and Denker, J., 1992, Multi-digit recognition using a space displacement neural network, in *Advances in Neural Information Processing Systems 4* (J. Moody, S. Hanson, and R. Lipmann, Eds.), San Mateo, CA: Morgan Kaufmann, pp. 488–495.

Mozer, M., 1991, *The Perception of Multiple Objects, A Connectionist Approach*, Cambridge, MA: MIT Press.

Rumelhart, D., Hinton, G., and Williams, R., 1986, Learning representations by back-propagating errors, *Nature*, 323:533–536.

Schenkel, M., Weissman, H., Guyon, I., Nohl, C., and Henderson, D., 1993, Recognition-based segmentation of on-line hand-printed words, in *Advances in Neural Information Processing Systems 5* (C. Hanson and L. Giles, Eds.), San Mateo, CA: Morgan Kaufmann, pp. 723–730.

Simard, P., and LeCun, Y., 1992, Reverse TDNN: An architecture for trajectory generation, in *Advances in Neural Information Processing Systems 4* (J. Moody, S. Hanson, and R. Lipmann, Eds.), San Mateo, CA: Morgan Kaufmann, pp. 579–588.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K., 1989, Phoneme recognition using time-delay neural networks, *IEEE Trans. Acoustics, Speech, Signal Processing*, 37:328–339.

Wolf, R., and Platt, J., 1994, Postal address block location using a convolutional locator network, in *Advances in Neural Information Processing Systems 6* (J. Cowan, G. Tesauro, and J. Alspector, Eds.), San Mateo, CA: Morgan Kaufmann, pp. 745–752.

Cooperative Behavior in Networks of Chaotic Elements

Kunihiko Kaneko

Chaos in the Brain

The importance of chaotic dynamics in the brain has been appreciated both theoretically and experimentally. Chaos is an irregular motion appearing in a deterministic system governed, for example, by some differential equations or maps. The necessity of chaotic dynamics in the brain can be inferred in the following contexts. So far it is the only scientific mechanism to connect deterministic and probabilistic views at a macroscopic level. By chaotic dynamics, a tiny perturbation can be amplified to a macroscopic level. Second, chaos is the only mechanism so far to create complexity from simple rules. For other examples of complex behavior, such as the spin glass model or cellular automaton, we need a huge number of degrees of freedom or rules (often made from "random couplings," externally assigned, which also require much information). The necessity of chaotic dynamics for biological information processing has been discussed by Tsuda (1992).

Experimentally, the existence of chaotic dynamics in the brain has been confirmed. Even a single neuron or a small ensemble of neurons can show a chaotic time series or frequency locking (Hayashi, Nakao, and Hirakawa, 1982). Some data of EEG time series show irregular motion, which is found to be chaotic with the aid of reconstruction of attractors or dimension estimates (Basar, 1990; Freeman, 1986). Partial synchronization of nonlinear oscillations has been discovered in the visual cortex (Gray et al., 1989; Eckhorn, 1991; see also SYNCHRONIZATION OF NEURONAL RESPONSES AS A PUTATIVE BINDING MECHANISM). Although the nature of their nonlinear oscillation has not yet been clarified, the dynamics strongly suggests the existence of chaos (but see CHAOS IN NEURAL SYSTEMS). Indeed, the significance of chaos can be deduced there; both the synchronization

connected first layer with, say, a few hundred hidden units, would already contain tens of thousands of weights. Overfitting problems may occur if training data are scarce. In addition, the memory requirement for that many weights may rule out certain hardware implementations. But the main deficiency of unstructured nets for image or speech aplications is that they have no built-in invariance with respect to translations, or local distortions of the inputs. Before being sent to the fixed-size input layer of a neural net, character images, spoken word spectra, or other 2D or 1D signals must be approximately size normalized and centered in the input field. Unfortunately, no such preprocessing can be perfect: handwriting is often normalized at the word level, which can cause size, slant, and position variations for individual characters; words can be spoken at varying speed, pitch, and intonation. This causes variations in the position of distinctive features in input objects. In principle, a fully connected network of sufficient size could learn to produce outputs that are invariant with respect to such variations. However, learning such a task would probably result in multiple units with identical weight patterns positioned at various locations in the input. Learning these weight configurations requires a very large number of training instances to cover the space of possible variations. Conversely, in convolutional networks (which will be defined in the next section), shift invariance is automatically obtained by forcing the replication of weight configurations across space.

Second, a deficiency of fully connected architectures is that the topology of the input is entirely ignored. The input variables can be presented in any (fixed) order without affecting the outcome of the training. On the contrary, images, or spectral representations of speech, have a strong 2D local structure, and time series have a strong 1D structure: variables (or pixels) that are spatially or temporally nearby are highly correlated. Local correlations are the reasons for the well-known advantages of extracting and combining *local* features before recognizing spatial or temporal objects. Convolutional networks force the extraction of local features by restricting the receptive fields of hidden units to be local.

Convolutional Networks

Convolutional networks combine three architectural ideas to ensure some degree of shift and distortion invariance: local receptive fields, shared weights (or weight replication), and, sometimes, spatial or temporal subsampling. A typical convolutional network for recognizing characters is shown in Figure 1 (from LeCun et al., 1990). The input plane receives images of characters that are approximately size normalized and centered. Each unit of a layer receives inputs from a set of units located in a small neighborhood in the previous layer. The idea of connecting units to local receptive fields on the input goes back to the perceptron in the early 1960s, and was almost simultaneous with Hubel and Wiesel's discovery of locally sensitive, orientation-selective neurons in the cat's visual system. Local connections have been reused many times in neural models of visual learning (see Mozer, 1991; LeCun, 1986; and NEOCOGNITRON). With local receptive fields, neurons can extract elementary visual features such as oriented edges, endpoints, or corners (or similar features in speech spectrograms). These features are then combined by the higher layers. As stated earlier, distortions or shifts of the input can cause the position of salient features to vary. In addition, elementary feature detectors that are useful on one part of the image are likely to be useful across the entire image. This knowledge can be applied by forcing a set of units, whose receptive fields are located at different places on the image, to have identical weight vectors (Rumelhart, Hinton, and Williams, 1986). The outputs of such a set of neurons constitute a *feature map*. At each position, different types of units in different feature maps compute different types of features. A sequential implementation of this, for each feature map, would be to scan the input image with a single neuron that has a local receptive field and to store the states of this neuron at corresponding locations in the feature map. This operation is equivalent to a convolution with a small-size kernel, followed by a squashing function. The process can be performed in parallel by implementing the feature map as a plane of neurons that *share* a single weight vector. Units in a feature map are constrained to perform the same operation on different parts of the image. A convolutional layer is usually composed of several feature maps (with different weight vectors), so that multiple features can be extracted at each location. The first hidden layer in Figure 1 has four feature maps with 5×5 receptive fields. Shifting the input of a convolutional layer will shift the output but will leave it unchanged otherwise. Once a feature has been detected, its exact location becomes less important, as long as its approximate position relative to other features is preserved. Therefore, each convolutional layer is followed by an additional layer which performs a local averaging and a subsampling, reducing the resolution of the feature map, and reducing the sensitivity of the output to shifts and distortions. The second hidden layer in Figure 1 performs 2×2 averaging and subsampling, followed by a trainable coefficient, a trainable bias, and a sigmoid. The trainable coefficient and bias control the effect of the squashing nonlinearity (for example, if the coefficient is small, then the neuron operates in a quasi-linear mode). Successive layers of convolutions and subsampling are typically alternated, resulting in a *bi-pyramid*: at each layer, the number of feature maps is increased as the spatial resolution is decreased. Each unit in the third hidden layer in Figure 1 may have input connections from several feature maps in the previous layer. The convolution/subsampling combination, inspired by Hubel and Wiesel's notions of "simple"

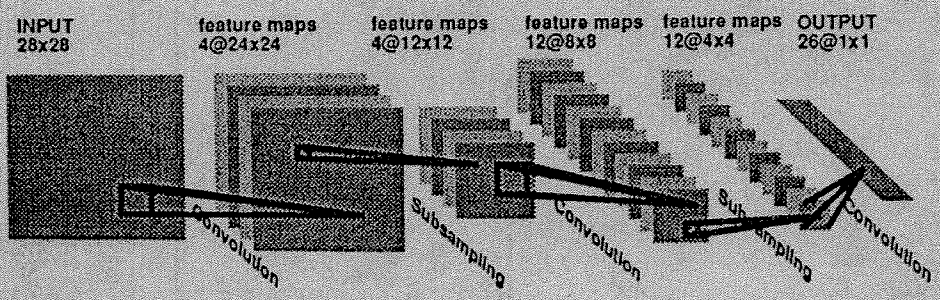

Figure 1. Convolutional neural network for image processing, e.g., handwriting recognition.

and "complex" cells, was implemented in the neocognitron model (see NEOCOGNITRON), although no globally supervised learning procedure such as backpropagation was available then.

Since all the weights are learned with backpropagation, convolutional networks can be seen as synthesizing their own feature extractor. The weight-sharing technique has the interesting side effect of reducing the number of free parameters, thereby reducing the "capacity" of the machine and improving its generalization ability (see LeCun, 1989, on weight sharing, and LEARNING AND GENERALIZATION: THEORETICAL BOUNDS for an explanation of generalization). The network in Figure 1 contains approximately 100,000 connections, but only approximately 2600 free parameters because of the weight sharing. Such networks compare favorably with other methods on handwritten character recognition tasks (Bottou et al., 1994; see also HANDWRITTEN DIGIT STRING RECOGNITION), and they have been deployed in commercial applications.

Fixed-size convolutional networks that share weights along a single temporal dimension are known as Time-Delay Neural Networks (TDNNs). TDNNs have been used in phoneme recognition (without subsampling) (Lang and Hinton, 1988; Waibel et al., 1989), spoken word recognition (with subsampling) (Bottou et al., 1990), and on-line handwriting recognition (Guyon et al., 1991).

Variable-Size Convolutional Networks: SDNNs

While characters or short spoken words can be size normalized and fed to a fixed-size network, more complex objects such as written or spoken words and sentences have inherently variable size. One way of handling such a composite object is to segment it heuristically into simpler objects that can be recognized individually (e.g., characters, phonemes). However, reliable segmentation heuristics do not exist for speech or cursive handwriting. A brute-force solution is to scan (or replicate) a recognizer at all possible locations across the input. While this can be prohibitively expensive in general, convolutional networks can be scanned or replicated very efficiently over large, variable-size input fields. Consider one instance of a convolutional net and its *alter ego* at a nearby location. Because of the convolutional nature of the networks, units in the two nets that look at identical locations on the input have identical outputs; therefore, their output does not need to be computed twice. In effect, replicating a convolutional network can be done simply by increasing the size of the field over which the convolutions are performed, and replicating the output layer, effectively making it a convolutional layer. An output whose receptive field is centered on an elementary object will produce the class of this object, while an in-between output may be empty or contain garbage. The outputs can be interpreted as evidence for the categories of object centered at different positions of the input field. A postprocessor therefore is required to pull out consistent interpretations of the output. Hidden Markov models (HMMs) or other graph-based methods are often used for that purpose (see SPEECH RECOGNITION: PATTERN MATCHING, and PATTERN RECOGNITION in this volume). The replicated network and the HMM can be trained simultaneously by backpropagating gradients through the HMM. Globally trained, variable-size TDNN/HMM hybrids have been used for speech recognition (see PATTERN RECOGNITION for a list of references) and on-line handwriting recognition (Schenkel et al., 1993). Two-dimensional replicated convolutional networks, called *Space Displacement Neural Networks* (SDNNs) (Figure 2), have been used in combination with HMMs or other elastic matching methods for handwritten word recognition (Keeler, Rumelhart, and Leow, 1991; Matan et al., 1992; Bengio, LeCun, and Henderson, 1994). Another interesting application of SDNNs is object spotting (Wolf and Platt, 1994).

An important advantage of convolutional neural networks is the ease with which they can be implemented in hardware. Specialized analog/digital chips have been designed and used in character recognition and in image-preprocessing applications (Boser et al., 1991). Speeds of more than 1000 characters per second were obtained with a network with approximately 100,000 connections.

The idea of subsampling can be turned around to construct networks that are similar to TDNNs but can generate sequences from labels. These networks are called reverse-TDNNs because they can be viewed as upside-down TDNNs: temporal resolution increases from the input to the output, through alternated oversampling and convolution layers (Simard and LeCun, 1992).

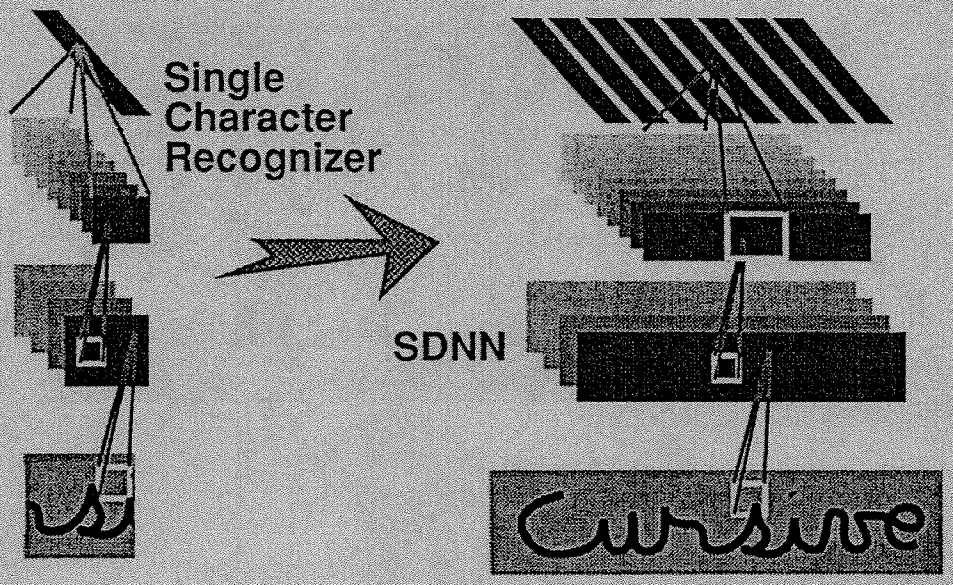

Figure 2. A variable-size replicated convolutional network, called a Space Displacement Neural Network (SDNN).